ESCAPE: Extensible Service ChAin Prototyping Environment using Mininet, Click, NETCONF and POX

Attila Csoma, Balázs Sonkoly*, Levente Csikor*, Felicián Németh†, András Gulyás†
Budapest Univ. of Technology and Economics‡
{csoma,sonkoly,csikor,nemethf,gulyas}@tmit.bme.hu

Wouter Tavernier, Sahel Sahhaf
Ghent University – iMinds, Ghent, Belgium
{wouter.tavernier,sahel.sahhaf}@intec.ugent.be

ABSTRACT

Mininet is a great prototyping tool which combines existing SDN-related software components (e.g., Open vSwitch, OpenFlow controllers, network namespaces, cgroups) into a framework, which can automatically set up and configure customized OpenFlow testbeds scaling up to hundreds of nodes. Standing on the shoulders of Mininet, we implement a similar prototyping system called ESCAPE, which can be used to develop and test various components of the service chaining architecture. Our framework incorporates Click for implementing Virtual Network Functions (VNF), NETCONF for managing Click-based VNFs and POX for taking care of traffic steering. We also add our extensible Orchestrator module, which can accommodate mapping algorithms from abstract service descriptions to deployed and running service chains.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network management*

Keywords

Service chain; Prototyping; SDN; Mininet; Click; NETCONF

1. INTRODUCTION

The concept of service chaining is not new but due to the networking revolution driven by SDN, it seems to re-magnetize the research/networking community [1]. A typical service consists of a series of service functions, traditionally implemented by middleboxes that have to be traversed in a given order by traffic flows. Service chain (or more generally service graph) is an abstraction to describe high level services in a generic way and to assemble processing flows for given traffic. Today's service deployment, service provisioning and service chaining have several limitations in terms of dynamics, flexibility, scalability, optimal usage of resources, as the built-in mechanisms are strongly coupled to the physical topology and the capabilities of special purpose, expensive hardware elements. As a consequence, configuring/deploying/operating service chains is a complex task and usually requires human interaction (the main problems are highlighted in [1]). In order to address these issues, different activities and research projects have been initiated. For instance, a dedicated working group (Service Function Chaining Working Group) has been established by the IETF dealing with several aspects of the service chaining architecture. The Network Functions Virtualization (NFV) group within ETSI aims at providing software-based telecommunication services, which can run in virtualized environment on a wide range of server hardwares instead of special purpose proprietary appliances. UNIFY (http://www.fp7-unify.eu) is an EU-funded FP7 project, which aims at unifying Cloud and carrier networks by developing an automated, dynamic service creation architecture based on a *dynamic fine-granular service chaining* model leveraging Cloud virtualization techniques and SDN.

In this demonstration, we present ESCAPE, our prototyping framework developed to the UNIFY architecture. Our system makes use of well-known, widely used tools, such as Mininet [2], Click [3], POX, and NETCONF, integrated into a common framework. An extensible Orchestrator module is also added, which makes our framework capable of setting up and configuring service chains on demand, mapping VNFs into physical resources, steering traffic according to chains' policies, and providing real-time management information on running VNFs.

2. ARCHITECTURE

The architecture proposed by UNIFY consists of three relevant layers enabling programmability at different abstraction levels. *Service layer* is aware of the service logic, handles service requests, and is responsible for SLAs. *Orchestrator layer* is the core of the framework. Based on a global network and resource view, it is responsible for mapping service requests to available resources and optimization. *Infrastructure layer* contains physical and virtual resources including compute, storage and networking resources, and components corresponding to local resource management. ESCAPE captures all these layers (Fig. 1) and provides a common plat-

*MTA-BME Future Internet Research Group
†MTA-BME Information systems research group
‡This work was conducted within the framework of the FP7 UNIFY project, which is partially funded by the Commission of the European Union.

SIGCOMM'14, August 17–22, 2014, Chicago, IL, USA.
ACM 978-1-4503-2836-4/14/08.
http://dx.doi.org/10.1145/2619239.2631448.

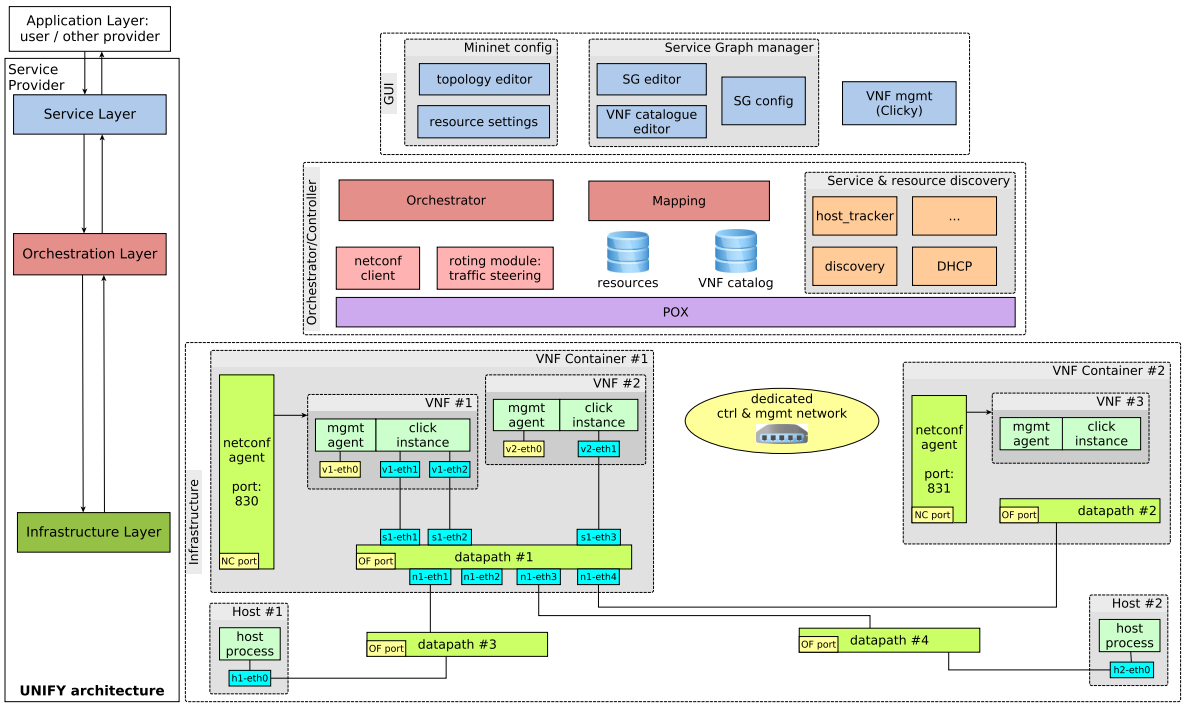

Figure 1: The main components of ESCAPE with the corresponding UNIFY architecture layers. The infrastructure layer is built on Mininet [2], which is a light-weight network emulation tool enabling agile prototyping.

form where the main steps of the service chaining process can be controlled, configured and further developed.

Our goal is to alleviate the developers' tedious task of setting up a whole service chaining environment and let them focus on their own work (e.g., developing a particular VNF, implementing an orchestration algorithm or a customized traffic steering). On the one hand, ESCAPE fosters VNF development by providing a simple, Mininet-based API where service graphs (SG), built from given VNFs, can be instantiated and tested automatically. ESCAPE also contains a VNF catalog, which is a built-in set of useful VNFs implemented in Click. The network infrastructure consists of OpenFlow switches (e.g., Open vSwitch) and a dedicated easy-to-configure controller application (implemented in the POX OpenFlow controller platform) is responsible for steering traffic between VNFs. Mininet is extended by the notion of VNFs that can be started as processes with configurable isolation models (based on cgroups in Linux). The establishment of dedicated control network is also possible where the management agents of the VNFs are connected to in order to be available from the service layer.

On the other hand, ESCAPE supports the development and the testing of orchestration components. Mininet is extended by NETCONF capability in order to support managed nodes (VNF containers) hosting VNFs. For this purpose, we integrate OpenYuma [4], an open-source NETCONF implementation, in the framework. A NETCONF agent is responsible for managing VNF containers and assigned switch(es). More specifically, the agent is able to start/stop VNFs and connect/disconnect VNFs to/from switches. The operation of the agent is described by the YANG data modeling language and implemented by low-level instrumentation codes. It is worth noting that the migration to real platforms require only the adaptation of the instrumentation part. The exposed RPCs of the agent are called from the orchestrator (NETCONF client module) to start/stop VNFs on demand. The paths are handled similarly by our traffic steering module. A dedicated component maps abstract service graphs into available resources based on different optimization algorithms (which can be easily changed or customized). Other components play a role in the automation of configuration processes. On top of these, a MiniEdit based GUI can be used to describe service graphs with given requirements (e.g., delay or bandwidth requirement on a sub-graph) and test topologies (resources and topology).

During the demo, we showcase every part of the architecture in a common GUI with the following steps[1]. The audience can (1) define VNF containers and the rest of the topology, (2) use the SG editor to create an abstract service graph where VNFs can be selected from a predefined list, (3) initiate the SG mapping to network resources and the deployment, (4) use standard tools to send and inspect live traffic, and (5) monitor the VNFs with Clicky.

3. REFERENCES

[1] P. Quinn and T. Nadeau. Service function chaining problem statement. IETF Draft, April 17, 2014.

[2] B. Lantz, B. Heller and N. McKeown. A network in a laptop: Rapid prototyping for software-defined networks. In *ACM HotNets 2010*.

[3] E. Kohler et al. The click modular router. *ACM Trans. Comput. Syst.*, 18(3):263–297, August 2000.

[4] OpenYuma. https://github.com/openclovis/openyuma.

[1] A screencast showing a simplified version of the demo is available: http://youtu.be/8ulstYXpHJA, http://sb.tmit.bme.hu/mediawiki/index.php/ESCAPE